Object-Relational Mapping Basics using Active Record (two talks)

qvr.qwizdom.com Q5VN94

Chris Loftus
cwl@aber.ac.uk

Overview

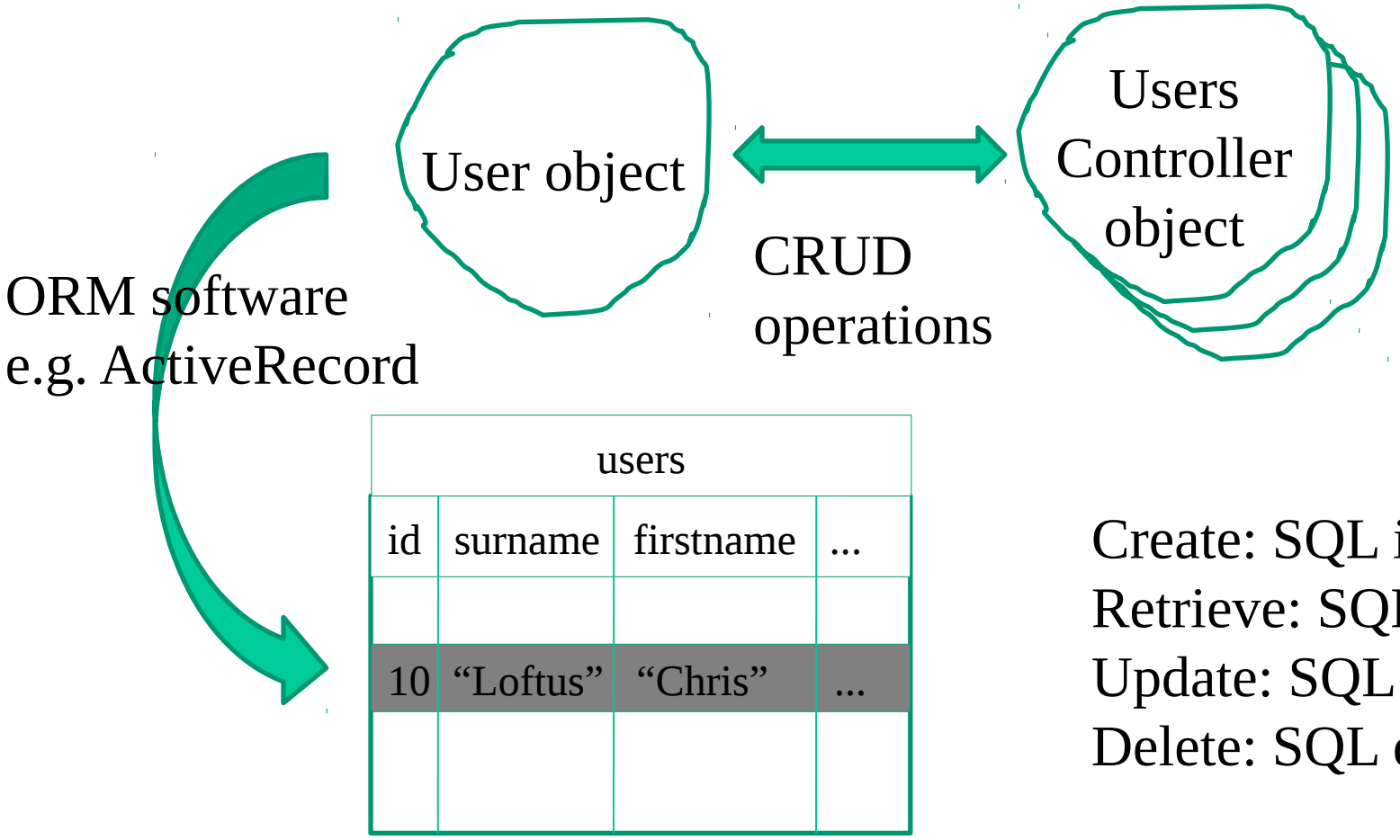
Talk 1

- Basic idea behind ORM
- Why use ORM?
- Active Record tables
- The mapping of database columns to model classes
- Mapping database types to Ruby types
- Primary keys

Talk 2

- CRUD operation support

Basic idea



Create: SQL insert
Retrieve: SQL select
Update: SQL update
Delete: SQL delete

Why use ORM?

- Allows developers to focus on “business” code using the object-oriented paradigm rather than SQL...
- Reduces the need to create special Data Access Object classes...
  - Provides the encapsulation mechanism that insulates the program from database specific APIs...

Why use ORM?

- An ORM framework is likely to be more robust and efficient than your code...
- Creation of reusable “business” objects which can be used with different RDBMSs...
- **Q. Any disadvantages? Spend a few minutes discussing with your colleagues and I’ll capture on the tablet**

Active Record tables

- Active Record imposes some unusual mapping strategies...
- Table names are pluralized by default...

Model class name	Table name
Customer	customers
LineItem	line_items
Person	people

- Can be overridden:

```
class Order < ActiveRecord::Base
  self.table_name = "O_1999"
end
```

- Inflection rules can also be specified in *config/initializers/inflections.rb*

inflections.rb 〉No Selection

```ruby
# Be sure to restart your server when you modify this file.

# Add new inflection rules using the following format. Inflections
# are locale specific, and you may define rules for as many different
# locales as you wish. All of these examples are active by default:
# ActiveSupport::Inflector.inflections(:en) do |inflect|
#   inflect.plural /^(ox)$/i, '\1en'
#   inflect.singular /^(ox)en/i, '\1'
#   inflect.irregular 'person', 'people'
#   inflect.uncountable %w( fish sheep )
# end

ActiveSupport::Inflector.inflections(:en) do |inflect|
  # A rule that keeps capitalisation on acronyms
  inflect.acronym 'API'
end
```

```ruby
namespace :api do
  resources :users
end
```

I used this for the REST web service example to allow me to use module: **class API::UsersController** instead of: **class Api::UsersController**

Columns and attributes

- Model class attributes are implicitly derived from schema column names!...
- Typically, migrations define the schema and therefore also model attributes...
- Justified as part of the DRY philosophy
  - But what about horrid legacy schemas?
  - Use façade columns:

```
class User < ApplicationRecord
  def family_name
    read_attribute(:surname)
  end
  def family_name=(name)
    write_attribute(:surname, name)
  end
end
```

Type mappings

SQL Type	Migration type (Ruby class)	SQL Type	Migration type (Ruby class)
int, integer, number	:integer (Fixnum, Bignum)	float, number	:float (Float)
decimal	:decimal (BigDecimal, Fixnum)	char, varchar, string, varchar2	:string (String)
date	:date (Date)	datetime time, date	:datetime (DateTime)
clob, text	:text (String)	time, date, datetime	:time (Time::Value)
blob, image, object, bytea	:binary (String)	boolean, decimal, tinyint, bit, number	:boolean (TrueClass, FalseClass)

For booleans values 0, “f”, “false”, “”, nil, false are interpreted as false

user.jobs? will always return boolean (false, true)

user.jobs_before_type_cast will return “f” or “t” for sqlite

Primary keys and identifiers

- Migrations, by default, add an *id* column that is used as the primary key...
- Justification: Domain data format sometimes gets changed, e.g. ISBN-10 to ISBN-13, which becomes painful if used as primary key
  - An integer id will be domain neutral...
- We can add indexes to our schema to maintain performance...
- What about legacy schemas?...
- Composites are not supported...

```
class User < ApplicationRecord
  self.primary_key = :email
end
...
user = User.new
user.id = "cwl@aber.ac.uk"
```

Muddy Points? Choose one option

A. I understood the topics presented
B. I am still unsure about the basic ORM concept
C. I am still unsure about the basic pros and cons of ORM
D. I am still unsure about the basic Active Record table and types mapping
E. I am still unsure about Active Record primary keys